4

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/728,543	12/01/2000	David Yach	555255012129	4943

7590 06/12/2007

David B. Cochran
Jones, Day, Reavis & Pogue
North Point
901 Lakeside Avenue
Cleveland, OH 44114

EXAMINER	
STRANGE, AARON N	
ART UNIT	PAPER NUMBER
2153	
MAIL DATE	DELIVERY MODE
06/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/728,543
Filing Date: December 01, 2000
Appellant(s): YACH, DAVID

David B. Cochran
Reg. No. 39,142
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2/27/2007 appealing from the Office action mailed 9/7/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is substantially correct. The changes are as follows:

WITHDRAWN REJECTIONS

The following grounds of rejection are not presented for review on appeal because they have been withdrawn by the examiner.

The rejection of claims 44-52, 55-61, 64-70, 73 and 74 under 35 U.S.C § 102(e) as being anticipated by Lowery (US 6,446,111), or in the alternative, under 35 U.S.C. § 103(a) as obvious over Lowery in view of Schwartz.

GROUNDS OF REJECTION NOT ON REVIEW

The following grounds of rejection have not been withdrawn by the examiner, but they are not under review on appeal because they have not been presented for review in the appellant's brief.

The rejection of claim 47 under 35 U.S.C. § 103(a) as being unpatentable over Schwartz et al. (US 6,473,609)

The rejection of claim 53, 62 and 71 under 35 U.S.C. 103(a) as being unpatentable over Schwartz et al. (US 6,473,609) in view of McGarvey (US 5,926,631).

The rejection of claims 54, 63 and 72 under 35 U.S.C. 103(a) as being unpatentable over Schwartz et al. (US 6,473,609) in view of Gosling (US 5,630,066).

In summary, the remaining grounds of rejection to be reviewed on appeal are:

The rejection of claims 44-46, 48-52, 55-61, 64-70, 73 and 74 under 35 U.S.C. § 102(e) as being anticipated by Schwartz et al. (US 6,473,609).

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Schwartz et al. (Schwartz)	US Patent No. 6,473,609	Oct. 29, 2002 Filed Sep. 14, 1998
Microsoft Computer Dictionary, Fifth Edition		2002

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 44-46, 48-52, 55-57, 58-61, 64-70,73, and 74 are rejected under 35 U.S.C. 102(e) as being anticipated by Schwartz et al. (US 6,473,609).

With regard to claim 44, Schwartz discloses a method of browsing content maintained in a page-rendered language without the use of a page-rendering browser application on a mobile communication device, comprising:

generating a request for content at the mobile communication device (Fig 1, 106) and transmitting the request to a gateway (Fig 1, 114) coupling the mobile communication device to a data network (Fig 1, 100) (requests for content are sent via gateway)(Col 5, Lines 57-61);

forwarding the content request from the gateway to a server on the data network where the content is stored in the page-rendered language (network server stores files in various page-rendered languages) (Col 8, Lines 58-67);

returning the requested page-rendered content from the server to the gateway (link server receives messages from network server)(Col 8, Lines 48-52);

translating the page-rendered content into a programmatic language and generating a corresponding executable program at the gateway (page is translated into screen commands which are placed in a SDD binary)(Col 9, Lines 27-47 and Col 10, Lines 3-8); and

forwarding the executable program to the mobile communication device which executes the program in order to browse the requested content (Col 10, Lines 3-8).

With regard to claim 46, Schwartz further discloses providing a byte code generator at the gateway; and executing the byte code generator to compress the translated programmatic language into byte codes (screen commands are compressed into byte codes) (Col 9, lines 39-47).

With regard to claim 48, Schwartz further discloses that the page-rendered language is HTML, HDML, XML or WML (Col 8, Lines 62-67).

With regard to claim 49, Schwartz further discloses that the page-rendered content returned from the server to the gateway is one of a plurality of content types, the method further comprising:

providing a plurality of content translators at the gateway, each of the plurality of content translators translating page-rendered content of a particular content type into a

common type of programmatic language (several different types can be converted) (Co 8, Lines 62-67);

determining the content type of the page-rendered content returned from the server to the gateway (Col 9, Lines 1-4); and

selecting and executing one of the plurality of content translators (HDML translator) at the gateway in correspondence with the determined type of content in order to translate the page-rendered content into the programmatic language (message is analyzed and converted)(Col 9, Lines 31-36).

With regard to claim 50, Schwartz further discloses that the mobile communication device is coupled to the gateway via a wireless data network (Fig 1, 102).

With regard to claim 51, Schwartz further discloses that the programmatic language is a virtual machine language (SDD screen command), the method further comprising:

providing a virtual machine (interface engine) at the mobile communication device for executing programs coded in the virtual machine language (Col 10, Lines 3-8).

With regard to claim 52, Schwartz further discloses providing a file explorer at the mobile communication device (Col 9, Lines 1-5), the file explorer generating the request

for content and storing the executable program forwarded from the gateway (Col 10, Lines 3-17).

Claims 55-57 and 58-61 are rejected under the same rationale as claims 44-46 and 48-52, respectively, since they recite substantially identical subject matter. Any differences between the claims do not result in patentably distinct claims and all of the limitations are taught by the above cited art.

Claims 64,67,68,66, and 69 are rejected under the same rationale as claims 44,45,46,48, and 51, respectively, since they recite substantially identical subject matter. Any differences between the claims do not result in patentably distinct claims and all of the limitations are taught by the above cited art.

Claim 65 is rejected under the same rationale as claims 44, 51, and 52, since claim 65 is substantially the combination of those claims. Any differences between the claims do not result in patentably distinct claims and all of the limitations are taught by the above cited art.

Claim 70 is rejected under the same rationale as claim 51, since they recite substantially identical subject matter. Any differences between the claims do not result in patentably distinct claims and all of the limitations are taught by the above cited art.

Claim 73 is rejected under the same rationale as claim 44, since they recite substantially identical subject matter. Any differences between the claims do not result in patentably distinct claims all of the limitations are taught by the above cited art.

Claim 74 is rejected under the same rationale as claims 44,48, and 51 since claim 74 is substantially the combination of those claims. Any differences between the claims do not result in patentably distinct claims and all of the limitations are taught by the above cited art.

(10) Response to Argument

A summary of the various points raised by Appellant is presented below, and each point is addressed individually by the Examiner.

Regarding claims 44, 55, 64, 65, 73 and 74:

a) Appellant argues that Schwartz provides an example "ASCII-like" representation of an SDD file that clearly shows that it is a data file formatted in a page-rendered language (Page 10 of the Brief).

b) Appellant argues that Schwartz fails to disclose converting page-rendered code into programmatic code that can be directly executed by the client device (Pages 9-10 of the Brief).

In reply to argument (a) that Schwartz provides an example "ASCII-like" representation of an SDD file that clearly shows that it is a data file formatted in a page-rendered language, the Examiner respectfully disagrees.

The cited portion of Schwartz (Col 9, Lines 50-58) contains a *representation* of an SDD file. This is not the SDD file itself, but merely a representation to make it easier for the reader to understand what the device will display. The actual SDD file is transmitted in "SDD format", which is typically binary (Col 10, Lines 3-6). Binary files are known in the art, and they contain data specifically intended to be read by a computer.

An exemplary definition from the Microsoft Computer Dictionary has been provided, which defines a binary file as "A file consisting of a sequence of 8-bit data or executable code, as distinguished from files consisting of human-readable ASCII text. Binary files are usually in a form readable only by a program, often compressed or structured in a way that is easy for a particular program to read".

Therefore, it is clear that the "ASCII-like" representation cited by Appellant is not the SDD file cited by the Examiner, and that the SDD file contains screen commands in binary form that are intended to be read and executed in the interface engine.

In reply to argument (b) that Schwartz fails to disclose converting page-rendered code into programmatic code that can be directly executed by the client device, the Examiner respectfully disagrees.

While similar to the arguments discussed above in reply to argument (a), the Examiner would like to further explain how the SDD file taught by Schwartz anticipates the claimed "executable program".

It is not disputed that Schwartz discloses a gateway that received page-rendered content from a server (at least Col 8, Lines 46-67). Schwartz takes the page rendered content and translates (converts) it into a programmatic language (screen *commands*) and generates a corresponding executable program (SDD file contains screen *commands* and is in binary format. The interface engine executes it by performing the commands) (at least Col 9, Lines 31-36 and Col 10, Lines 3-8).

It is clear that a *binary* file containing commands that cause the mobile device to display certain items is an "executable program", well within the broadest reasonable interpretation of that term. Furthermore, even if the SDD file could not reasonable be interpreted as an executable program, which it can, Schwartz also discloses that the message processor can convert the received page rendered content into various *compiled* formats (at least Col 10, Lines 10-17). It is well known in the art that a computer can execute compiled code.

Additionally, it should be noted that the term "executable program", when interpreted in light of the specification, is clearly intended to encompass "interpreted" (Page 6, Lines 15-18 of the specification) virtual machine language programs such as those written in Java (at least Pages 13-17 of the specification) (also see claims 51, 60 and 69). It is well known in the art that virtual machine language programs such as Java are received by the virtual machine in binary data format (bytecodes) and interpreted by

the machine. For additional information, attention should be directed to the Microsoft Computer Dictionary, which defines the terms "Java" and "Java Virtual Machine". These programs are not "executed" directly by a computer in the traditional sense, but interpreted by the virtual machine operating on the computer to cause the virtual machine to produce output. This operation is substantially identical to the "rendering" performed by the interface engine in Schwartz.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

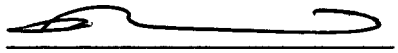
For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Aaron Strange

June 4, 2007

Conferees:



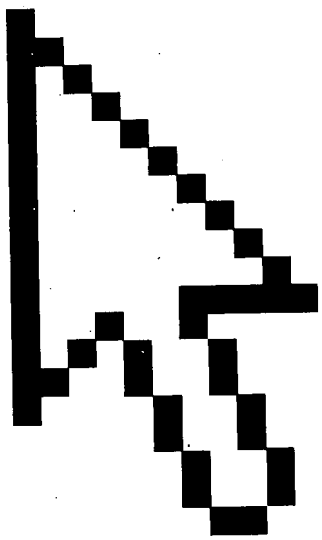
Lynne H Browne
Appeal Practice Specialist, TQAS
Technology Center 2100

Glenton Burgess
Supervisory Patent Examiner
Art Unit 2153

Microsoft

Microsoft Computer Dictionary

Fifth Edition



PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21

200219714

Printed and bound in the United States of America.

2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com*.

Acquisitions Editor: Alex Blanton
Project Editor: Sandra Haynes

Body Part No. X08-41929

and 0s. To ease translation, programmers and others who habitually work with the computer's internal processing abilities use hexadecimal (base-16) or octal (base-8) numbers. See Appendix E. *See also* base (definition 2), binary-coded decimal, binary number, bit, Boolean algebra, byte, cyclic binary code, digital computer, dyadic, logic circuit. *Compare* ASCII, decimal, hexadecimal, octal.

binary[2] *n.* In an FTP client program, the command that instructs the FTP server to send or receive files as binary data. *See also* FTP client, FTP server. *Compare* ascii.

binary chop *n. See* binary search.

binary-coded decimal *n.* A system for encoding decimal numbers in binary form to avoid rounding and conversion errors. In binary-coded decimal coding, each digit of a decimal number is coded separately as a binary numeral. Each of the decimal digits 0 through 9 is coded in 4 bits, and for ease of reading, each group of 4 bits is separated by a space. This format is also called 8-4-2-1, after the weights of the four bit positions, and uses the following codes: 0000 = 0; 0001 = 1; 0010 = 2; 0011 = 3; 0100 = 4; 0101 = 5; 0110 = 6; 0111 = 7; 1000 = 8; 1001 = 9. Thus, the decimal number 12 is 0001 0010 in binary-coded decimal notation. *Acronym:* BCD. *See also* base (definition 2), binary[1], binary number, decimal, EBCDIC, packed decimal, round.

binary compatibility *n.* Portability of executable programs (binary files) from one platform, or flavor of operating system, to another. *See also* flavor, portable (definition 1).

binary conversion *n.* The conversion of a number to or from the binary number system. See Appendix E. *See also* binary[1].

binary device *n.* Any device that processes information as a series of on/off or high/low electrical states. *See also* binary[1].

binary digit *n.* Either of the two digits in the binary number system, 0 and 1. *See also* bit.

binary file *n.* A file consisting of a sequence of 8-bit data or executable code, as distinguished from files consisting of human-readable ASCII text. Binary files are usually in a form readable only by a program, often compressed or structured in a way that is easy for a particular program to read. *Compare* ASCII file.

binary file transfer *n.* Transfer of a file containing arbitrary bytes or words, as opposed to a text file containing only printable characters (for example, ASCII characters with codes 10, 13, and 32–126). On modern operating systems a text file is simply a binary file that happens to contain only printable characters, but some older systems distinguish the two file types, requiring programs to handle them differently. *Acronym:* BFT.

binary format *n.* Any format that structures data in 8-bit form. Binary format is generally used to represent object code (program instructions translated into a machine-readable form) or data in a transmission stream. *See also* binary file.

binary notation *n.* Representation of numbers using the binary digits, 0 and 1. *Compare* floating-point notation.

binary number *n.* A number expressed in binary form, or base 2. Binary numbers are composed of zeros and ones. See Appendix E. *See also* binary[1].

binary search *n.* A type of search algorithm that seeks an item, with a known name, in an ordered list by first comparing the sought item to the item at the middle of the list's order. The search then divides the list in two, determines in which half of the order the item should be, and repeats this process until the sought item is found. *Also called:* binary chop, dichotomizing search. *See also* search algorithm. *Compare* hash search, linear search.

binary synchronous protocol *n. See* BISYNC.

binary transfer *n.* The preferred mode of electronic exchange for executable files, application data files, and encrypted files. *Compare* ASCII transfer.

binary tree *n.* In programming, a specific type of tree data structure in which each node has at most two subtrees, one left and one right. Binary trees are often used for sorting information; each node of the binary search tree contains a key, with values less than that key added to one subtree and values greater than that key added to the other. See the illustration. *See also* binary search, tree.

J

J2EE *n.* Acronym for **J**ava **2** Platform **E**nterprise **E**dition. An application server framework from Sun Microsystems, Inc., for the development of distributed applications. It includes all the previous Java APIs targeted for multitiered distributed enterprise information systems. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications. *See also* application programming interface, Enterprise Java Beans, IDL, Java, JDBC, Jini, JMS, JNDI, JSP, JTA, JTS, RMI-IIOP.

J *n.* A high-level programming language created by Kenneth Iverson, developer of APL, and Roger Hui. J is a successor language to APL that may be run on many platforms, including Windows 95, Windows NT, Macintosh, Linux, RS/6000, and Sun Sparc. Like APL, J is used primarily by mathematicians. *See also* APL.

jabber *n.* A continuous stream of random data transmitted over a network as the result of some malfunction.

Jabber *n.* An XML-based instant messaging system. Jabber software is available for most operating systems and allows user access to other instant messaging services. Jabber is an open source application overseen by Jabber.org.

jack *n.* A connector designed to receive a plug. A jack is commonly used in making audio and video connections.

jacket *n. See* disk jacket.

jack in *vb.* **1.** To log on to a computer. **2.** To connect to a network or BBS, especially for purposes of entering an IRC or a virtual reality simulation, such as a MUD. (To leave is to *jack out.*) *See also* IRC, MUD.

jack out *vb.* **1.** To log off a computer. **2.** To disconnect from a network or online bulletin board system. *See also* jack in, log on.

Jacquard loom *n.* The first machine that used punched cards to control its operation. In this loom, developed in 1801 by French inventor Joseph-Marie Jacquard, up to 24,000 cards were placed on a rolling drum. Where a hole was punched on a card, one of a set of rods could pass through and select a particular thread to be woven into the pattern. Jacquard was awarded a medal by the Emperor Napoleon for his invention. Later in the nineteenth century, punched cards were used in Charles Babbage's computerlike Analytical Engine and in Herman Hollerith's statistical tabulating machine. *See also* Analytical Engine, Hollerith tabulating/recording machine.

jaggies *n.* The "stairsteps" that appear in diagonal lines and curves drawn at low resolutions in computer graphics. *Also called:* aliasing.

Janet *n.* Short for the **J**oint **A**cademic **Net**work. A wide area network in the United Kingdom that serves as the principal backbone for the Internet in that country. *See also* backbone (definition 1).

.jar *n.* A file name extension that identifies a compressed JAR (**J**ava **Ar**chive) file. Note: By changing the .jar extension to .zip, you can use popular extraction tools such as PKZIP or WINZIP to look at a .jar file's contents. *See also* compressed file, JAR, PKZIP, .zip.

JAR *n.* Acronym for **J**ava **Ar**chive file. JAR files allow Java developers to efficiently deploy Java classes and their associated resources. The elements in a JAR file are compressed just as in a standard zip file. JAR files include a security mechanism and a special META-INF directory that contains administrative information about the contents of the files. Using a combination of a digital signature and the META-INF data, JAR files can be signed to ensure authenticity and security. *See also* .jar.

Java *n.* An object-oriented programming language developed by Sun Microsystems, Inc. Similar to C++, Java is smaller, more portable, and easier to use than C++ because it is more robust and it manages memory on its own. Java was also designed to be secure and platform-neutral (meaning that it can be run on any platform) through the fact that Java programs are compiled into bytecode, which is not refined to the point of relying on platform-specific instructions and runs on a computer in a special software environment known as a virtual machine. This characteristic of Java makes it a useful language for programming

Web applications, since users access the Web from many types of computers. Java is used in programming small applications, or applets, for the World Wide Web, as well as in creating distributed network applications. *See also* bytecode, Java applet, Jini, object-oriented programming.

Java applet *n.* A Java class that is loaded and run by an already-running Java application such as a Web browser or an applet viewer. Java applets can be downloaded and run by any Web browser capable of interpreting Java, such as Internet Explorer, Netscape Navigator, and HotJava. Java applets are frequently used to add multimedia effects and interactivity to Web pages, such as background music, real-time video displays, animations, calculators, and interactive games. Applets can be activated automatically when a user views a page, or they may require some action on the part of the user, such as clicking on an icon in the Web page. *See also* applet, Java.

JavaBean *n.* A Java component architecture defined in the JavaBeans specification developed by Sun Microsystems. A JavaBean, or Bean, is a reusable application component—an independent code segment—that can be combined with other JavaBean components to create a Java applet or application. The JavaBean concept emphasizes the platform-independence of the Java language, in which ideally a program, once written, can run on any computing platform. JavaBeans are similar to Microsoft's ActiveX controls. ActiveX controls, however, can be developed in different programming languages but executed only on a Windows platform. JavaBeans can be developed only in the Java programming language but ideally can run on any platform. *See also* ActiveX, Java.

Java Card *n.* An application programming interface (API) from Sun Microsystems, Inc., that allows Java applets and programs to run on smart cards and other devices with limited memory. Java Card uses a Java Card Virtual Machine designed for severely memory-constrained devices. *See also* applets, Java Card Virtual Machine, smart card (definition 2).

Java Card Virtual Machine *n.* An ultra-small-footprint, highly optimized foundation of a runtime environment within the Java 2 Platform Micro Edition. Derived from the Java Virtual Machine (JVM), it is targeted at smart cards and other severely memory-constrained devices. The Java Card Virtual Machine can run in devices with memory as small as 24 KB of ROM, 16 KB of EEPROM, and 512 bytes of RAM. *See also* EEPROM, Java Card, RAM, ROM.

Java chip *n.* An implementation on a single integrated circuit of the virtual machine specified for execution of the Java programming language. Such chips, which are being developed by Sun Microsystems, Inc., could be used in very small devices and as controllers for appliances. *See also* integrated circuit, Java, virtual machine.

Java-compliant browser *n.* A Web browser with support for the Java programming language built into it. Most current Web browsers are Java-compliant. *See also* Java, Web browser.

Java Developer's Kit *n.* A set of software tools developed by Sun Microsystems, Inc., for writing Java applets or applications. The kit, which is distributed free, includes a Java compiler, interpreter, debugger, viewer for applets, and documentation. *Acronym:* JDK. *See also* applet, Java, Java applet.

Java Foundation Classes *n.* A Java-based set of class libraries developed by Sun Microsystems, Inc. Encompassing fundamentals of the Internet Foundation Classes created by Netscape Communications Corp., the Java Foundation Classes extend the Java Abstract Window Toolkit (AWT) by providing graphical user interface components for use in developing commercial and Internet-related Java applications. *See also* Abstract Window Toolkit, Application Foundation Classes, Internet Foundation Classes, Java, JavaBean, Microsoft Foundation Classes.

Java HotSpot *n.* A Java performance engine introduced by Sun Microsystems, Inc., in 1999 that is designed to run Java applications faster than just-in-time (JIT) compilers. The core of Java HotSpot, and the feature for which it is named, is its ability to perform adaptive optimization—the identification and optimization of "hot spots," or sections of performance-critical code. Improved garbage collection (freeing of memory occupied by objects no longer in use) and better multithreading are additional features designed to contribute to increased performance. *See also* Java.

Java IDL *n.* Short for **Java** Interface Definition Language. A Java technology that provides CORBA interoperability and connectivity capabilities for the Java platform. These capabilities enable Java applications to invoke operations on remote network services using the Object Management Group Interface Definition Language and Internet Inter-ORB Protocol. *See also* CORBA, IDL, J2EE, RMI-IIOP.

JavaMail *n.* An API in the Sun Microsystems, Inc., Java platform for sending and receiving mail. A set of abstract APIs that model a mail system, JavaMail provides a platform-independent and protocol-independent

framework to build Java-based e-mail client applications. *See also* application programming interface, e-mail, J2EE.

Java Management Application Programming Interface *n.* A set of application programming interface specifications, proposed by Sun Microsystems, Inc., to enable the Java language to be used for network management. *Acronym:* JMAPI. *See also* application programming interface, Java.

JavaOS *n.* An operating system designed to run applications written in the Java programming language. JavaOS was created by JavaSoft, an operating company of Sun Microsystems, Inc., to run the Java Virtual Machine (JVM) directly on microprocessors, and thus eliminate the need for a resident operating system. JavaOS is small and designed for network computers, as well as devices ranging from game machines to pagers and cellular telephones. *See also* Java.

JavaScript *n.* A scripting language developed by Netscape Communications and Sun Microsystems that is loosely related to Java. JavaScript, however, is not a true object-oriented language, and it is limited in performance compared with Java because it is not compiled. Basic online applications and functions can be added to Web pages with JavaScript, but the number and complexity of available application programming interface functions are fewer than those available with Java. JavaScript code, which is included in a Web page along with the HTML code, is generally considered easier to write than Java, especially for novice programmers. A JavaScript-compliant Web browser, such as Netscape Navigator or Internet Explorer, is necessary to run JavaScript code. *See also* application programming interface, HTML, scripting language. *Compare* Java.

JavaServer Pages *n. See* JSP.

Java Speech Grammar Format *n.* A platform-independent grammar description format developed for use with speech recognition systems. Java Speech Grammar Format is used extensively with Voice XML and can be used with most speech recognition systems and related applications. *Acronym:* JSGF.

Java Virtual Machine *n.* The environment in which Java programs run. The Java Virtual Machine gives Java programs a software-based "computer" they can interact with. (Programs, even the most seemingly unchallenging ones designed for children or entertainment, must run within an environment from which they can use memory, display information, gather input, and so on.) Because the Java Virtual Machine is not a real computer but exists in software, a Java program can run on any physical computing platform, such as a Windows 9x computer or a Macintosh, equipped with an interpreter—usually an Internet browser—that can carry out the program's instructions and a Java Virtual Machine that provides the "hardware" on which the program can run. *Acronym:* JVM.

JCL *n.* Acronym for **J**ob **C**ontrol **L**anguage. A command language used in IBM OS/360 mainframe systems. JCL is used to launch applications and specifies information on running time, program size, and the program files used for each application. *See also* command language.

JDBC *n.* A Java API designed to provide access to relational databases and other tabular material, such as spreadsheets and flat files. Using JDBC, a developer can create a cross-platform Java application that can connect with, and send SQL statements to, a number of different relational databases. Although it is commonly thought to stand for Java Database Connectivity, JDBC is the name of the technology; it is not an acronym.

JDK *n. See* Java Developer's Kit.

JDoc *n.* A cross-platform, interactive format for display, distribution, and interaction with live Web pages. jDoc documents are small in size and can be embedded in HTML documents to offer client-side interactivity. jDoc was created by EarthStones and is an extension to Sun's Java platform.

JetSend Protocol *n.* A platform-independent communications protocol developed by Hewlett-Packard to enable direct device-to-device communication. The JetSend protocol is designed to provide JetSend-enabled devices with the ability to exchange information and data without the need for device drivers or reliance on servers or user intervention. The protocol is intended for use with printers, scanners, fax machines, and other such information "appliances" and was developed to simplify and improve interoperability between and among a wide range of devices.

Jet SQL *n.* A query language. Jet SQL is a dialect used by the Microsoft Access application, specifically by the Microsoft Jet database engine, to extract, manipulate, and structure data that resides in a relational database management system (RDBMS). Jet SQL is based largely on the ANSI SQL-92 standard, with additional extensions.